Information Sciences 399 (2017) 13–29

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Differential evolution powered by collective information

Li Ming Zheng[a], Sheng Xin Zhang[a], Kit Sang Tang[b], Shao Yong Zheng[c,*]

[a] Department of Electronic Engineering, School of Information Science and Technology, Jinan University, Guangzhou 510632, China
[b] Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong
[c] School of Electronics and Information Technology, Sun Yat-sen University, No. 132 Waihuan East Road, Higher Education Mega Centre, Guangzhou 510006, China

ARTICLE INFO

Article history:
Received 14 June 2016
Revised 19 February 2017
Accepted 26 February 2017
Available online 3 March 2017

Keywords:
Differential evolution
Mutation
Crossover
Collective information

ABSTRACT

Differential evolution (DE) algorithms have demonstrated excellence performance in dealing with global optimization problems. In DE, mutation is the sole process providing new components to form potential candidates, and it does so by combining various existing solution vectors. In the past two decades, many mutation strategies have been proposed with the goal of achieving better searching capability. Commonly, the best candidate in the current population or its subset is employed. In this study, we challenge the approach of adopting only the single best vector and suggest enhancing DE with the collective information of the m best candidates. The evolutionary information of these m best candidates is linearly combined to form a part of the difference vector in mutation. Moreover, the collective information can also be used in crossover. Consequently, a new DE variant called collective information-powered differential evolution (CIPDE) is constructed. To verify its effectiveness, CIPDE is compared with seven state-of-the-art DE variants on 28 CEC2013 benchmark functions. Numerical results confirm that CIPDE is superior to the other DEs for most of the test functions. The impacts of the components of CIPDE and performance sensitivities to system parameters are also investigated.

1. Introduction

Differential evolution (DE) is a simple but powerful evolutionary algorithm (EA) proposed by Storn and Price [32]. Similar to other EAs, it utilizes a population-based stochastic search method instead of complex mathematical operations. Over the past two decades, DE has received much attention from diverse domains of scientific and engineering research. Not only have other advanced variants [6,7,15,26,28,31,38,43] evolved for different optimization problems, but DE has also been successfully applied to various real-world problems, such as electromagnetics [29], optics [49], pattern recognition [37] and signal processing [2].

In DE, the key effort to explore new components for potential solutions relies on the mutation process. It facilitates the search by providing a mutant vector, which is used in the next process (called crossover) for the generation of solution candidates. In most mutation strategies, the ingredients of the mutant vector are based on various randomly picked vector solutions or the best one in the current population. In some advanced DE variants [16,47,48], the choice of vector may extend to one from a group of top ranked vectors or the best group from a randomly selected subset of the population.

* Corresponding author.
E-mail address: zhengshaoy@mail.sysu.edu.cn (S.Y. Zheng).

http://dx.doi.org/10.1016/j.ins.2017.02.055
0020-0255/

As reflected by those advanced DE variants, a high-quality vector (in terms of fitness) is important, as it provides a preferable direction for the search. It is also common for only a single vector of this type to be employed. This is different from what is witnessed in other engineering aspects, such as human swarming [24] and IQ Social (IQS) [34], in which a significant impact of collective intelligence is observed. It is thus our motivation to study how collective information can impact DE design.

Alternately, DE suffers from the problem of stagnation, which deteriorates its performance [17]. When stagnation occurs, no better solution can be found from the newly created candidate solutions, even though the population remains diverse. As noted in [17], the occurrence probability of stagnation depends on the number of different potential trial vectors available and their chances of survival in the following generations.

In this paper, we propose a novel mutation strategy (CIM) that takes advantage of the collective information from the m best vectors in the population, where m ($m \in [1, i]$) is a random integer governed by the target vector of rank i. A combinational vector is formed by weighted contributions from these m best vectors and then used as a component to generate a mutant vector. This combinational vector is useful not only in mutation but also in crossover. We thus design a collective information-based crossover (CIX) that can help to relieve the problem of stagnation. Based on the proposed CIM and CIX operators, a collective information-powered DE variant named CIPDE is proposed. Detailed analyses are performed by evaluating its performance on 28 benchmark functions developed for the 2013 IEEE Congress on Evolutionary Computation (IEEE CEC2013) [19].

Our contributions are multifold. We are the first to propose the use of collective information from multiple best vectors in mutation and crossover. Moreover, based on the proposed mutation and crossover operations, we design a new DE variant, which achieves a better solution, as confirmed with the use of the CEC2013 test suite.

The rest of this paper is organized as follows. In Section 2, an overview of DE algorithms and an introduction to collective intelligence are presented. In Section 3, the proposed operations, CIM and CIX, together with the new DE variant, CIPDE, are described in detail. Extensive experiments were carried out; the results are presented in Section 4, where detailed comparison and analyses are also given. Finally, the study's conclusions are given in Section 5.

2. Overview of differential evolution

2.1. Basis

DE provides a direct search method for continuous optimization problems. It consists of four basic components: initialization, mutation, crossover and selection.

Initialization: Consider a D-dimensional problem; an initial population $P_0 = \{\vec{x}_{i,0} = (x_{i,1,0}, x_{i,2,0}, \ldots, x_{i,D,0}), i \in \Phi \equiv \{1, 2, \ldots, NP\}\}$ is randomly generated within the search domain, where NP is the population size. Moreover, other user-defined operational parameters are specified.

After the initialization, DE enters a simple loop of mutation, crossover and selection until some specified stopping criteria are met.

Mutation: Its function is to generate a mutant vector $\vec{v}_{i,G}$ corresponding to each ith vector, with $i = 1, 2, \ldots, NP$, in the current population P_G, where G is the generation. Some widely used mutation strategies in the literature include:

$$\text{rand}/1: \quad \vec{v}_{i,G} = \vec{x}_{r_1^i,G} + F \cdot (\vec{x}_{r_2^i,G} - \vec{x}_{r_3^i,G}) \tag{1}$$

$$\text{best}/1: \quad \vec{v}_{i,G} = \vec{x}_{best,G} + F \cdot (\vec{x}_{r_1^i,G} - \vec{x}_{r_2^i,G}) \tag{2}$$

$$\text{current} - \text{to} - \text{best}/1: \quad \vec{v}_{i,G} = \vec{x}_{i,G} + F \cdot (\vec{x}_{best,G} - \vec{x}_{i,G}) + F \cdot (\vec{x}_{r_1^i,G} - \vec{x}_{r_2^i,G}) \tag{3}$$

$$\text{current} - \text{to} - \text{rand}/1: \quad \vec{v}_{i,G} = \vec{x}_{i,G} + K \cdot (\vec{x}_{r_1^i,G} - \vec{x}_{i,G}) + F \cdot (\vec{x}_{r_2^i,G} - \vec{x}_{r_3^i,G}) \tag{4}$$

where the distinct r_j^i indices are randomly chosen from $\Phi \backslash \{i\}$, $\vec{x}_{i,G}$ is the target vector, $\vec{x}_{best,G}$ is the best vector in P_G, and F and K are two user-defined mutation factors within (0, 1].

Crossover: Its function is to generate a trial vector $\vec{u}_{i,G} = \{u_{i,1,G}, u_{i,2,G}, \ldots, u_{i,D,G}\}$ based on the target vector $\vec{x}_{i,G}$ and its corresponding mutant vector $\vec{v}_{i,G}$. The classic binomial crossover is defined as:

$$u_{i,j,G} = \begin{cases} v_{i,j,G} & \text{if } rand_j(0,1) \le CR \text{ or } j = j_{rand} \\ x_{i,j,G} & \text{otherwise} \end{cases} \tag{5}$$

where $rand_j$ (0, 1) is a randomly generated variable within (0, 1), CR is a user-defined crossover factor restricted to (0, 1], and j_{rand} is a uniformly distributed random integer within [1, D]. The condition $j = j_{\text{rand}}$ prevents a direct copy from $\vec{x}_{i,G}$ to $\vec{u}_{i,G}$.

Algorithm 1
Classic DE with the "rand/1" strategy (**DE**).

1: Set the population size *NP*, mutation factor *F*, and crossover factor *CR*; initialize and evaluate the population $\mathbf{P_G}$; set the generation counter $G = 0$;
2: **While** the stopping criteria are not met **Do**
3: **For** $i = 1$: *NP* **Do**
———— Classic mutation operation————-
4: Randomly choose $\vec{x}_{r_1^i,G}$, $\vec{x}_{r_2^i,G}$ and $\vec{x}_{r_3^i,G}$ from the current population $\mathbf{P_G}$, where $r_1 \neq r_2 \neq r_3 \neq i$; generate a mutant vector $\vec{v}_{i,G}$ using Eq. (1);
———— Classic crossover operation————-
5: Generate $j_{\text{rand}} = \text{randint}[1, D]$;
6: **For** $j = 1$: D **Do**
7: **If** $\text{rand}_j(0,1) \leq CR$ or $j = j_{\text{rand}}$, **then** $u_{i,j,G} = v_{i,j,G}$;
8: **Else** $u_{i,j,G} = x_{i,j,G}$;
9: **End If**
10: **End For**
————-Selection operation————-
11: Evaluate the fitness of $\vec{u}_{i,G}$;
12: **If** $f(\vec{u}_{i,G}) \leq f(\vec{x}_{i,G})$, **then** $\vec{x}_{i,G+1} = \vec{u}_{i,G}$;
13: **Else** $\vec{x}_{i,G+1} = \vec{x}_{i,G}$;
14: **End If**
15: **End For**
16: $G = G + 1$;
17: **End While**

Selection: The newly generated trial vectors are then evaluated and compared with the target vectors. The one with better fitness $f(\bullet)$ is selected for the next generation. In mathematics, one has

$$\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G} & \text{if } f(\vec{u}_{i,G}) \leq f(\vec{x}_{i,G}) \\ \vec{x}_{i,G} & \text{otherwise} \end{cases} \tag{6}$$

The general flow of DE is described by the pseudocode of the classic DE with the "DE/rand/1" strategy, as presented in Algorithm 1.

2.2. Improvements and modifications

Over the past two decades, various DE variants with enhanced performance have been proposed. The first common method of improvement is to design some new mutation strategy. In [48], by modifying the "current-to-best/1"approach, a new mutation strategy, denoted as "current-to-pbest/1," is proposed:

$$\vec{v}_{i,G} = \vec{x}_{i,G} + F \cdot (\vec{x}_{pbest^i,G} - \vec{x}_{i,G}) + F \cdot (\vec{x}_{r_1^i,G} - \vec{x}_{r_2^i,G}) \tag{7}$$

where $\vec{x}_{pbest^i,G}$ is randomly selected from the current top $100 \times p\%$ ($p \in (0, 1]$) best solutions. Another similar strategy, known as "current-to-gr_best/1," is presented in [16]. Instead of randomly selecting one of the top solutions as in (7), it uses the information from the best among a group of randomly selected solutions (denoted as $\vec{x}_{gr\_best^i,G}$); the corresponding operation is described as

$$\vec{v}_{i,G} = \vec{x}_{i,G} + F \cdot (\vec{x}_{gr\_best^i,G} - \vec{x}_{i,G}) + F \cdot (\vec{x}_{r_1^i,G} - \vec{x}_{r_2^i,G}) \tag{8}$$

In [10], "rand/1" is extended to a new trigonometric mutation operator, which is given by

$$\vec{v}_{i,G} = (\vec{x}_{r_1^i,G} + \vec{x}_{r_2^i,G} + \vec{x}_{r_3^i,G})/3 + (p_2 - p_1) \cdot (\vec{x}_{r_1^i,G} - \vec{x}_{r_2^i,G}) + (p_3 - p_2) \cdot (\vec{x}_{r_2^i,G} - \vec{x}_{r_3^i,G}) + (p_1 - p_3) \cdot (\vec{x}_{r_3^i,G} - \vec{x}_{r_1^i,G}) \tag{9}$$

where $\vec{x}_{r_1^i,G}$, $\vec{x}_{r_2^i,G}$ and $\vec{x}_{r_3^i,G}$ are three randomly selected vectors from the current population $\mathbf{P_G}$, $p_j = \frac{1}{p}|f(\vec{x}_{r_j^i,G})|$ for $j = 1, 2, 3$ and $p = \sum_{j=1}^{3} f(\vec{x}_{r_j^i,G})$.

Another method of advancement is to combine multiple mutation strategies. In [10], the trigonometric mutation operation (9) is combined with the classic "rand/1" strategy with a probability. In [42], a Gaussian distribution-based mutation scheme is hybridized with the classic "best/1" strategy. In [5] and [25], global and local mutation operators are combined using weighting factors. Sometimes, a specific strategy may also be selected from a pool of strategies based on probability [20] or previous experiences [28]. In some proposals, all elements in the pool of strategies are tried [38] or applied separately to different subpopulations [36, 45] to enrich the search.

Modifications of other aspects in mutation have also been proposed. Instead of using a fixed control parameter, *F* can be self-adapted and specified independently for each vector [3]. In [48], it is suggested that *F* can be governed by a Cauchy distribution with a location parameter determined by the success cases. In [21], *F* is adapted by adopting a Gaussian adaptation algorithm. In [35,8], *F* is adjusted according to success history and sinusoidal formulas, respectively.

The probability of a population vector being selected for the mutation strategy may depend on its distance from the mutant vector [9], its fitness value [13] or various measures of convergence and divergence [41].

In addition to the modification of mutation, enhancements in crossover have also been investigated. For example, to improve the crossover operation, linearly scalable exponential crossover [50], covariance matrix learning [40], orthogonal design [39], and hybrid linkage crossover [4] have been employed.

2.3. Stagnation of DE

Although significant progress has been made over the last two decades, problems are still observed in classic DE and most of its variants. Similar to other EAs, a typical problem is premature convergence. Generally, it is caused by the imbalance of exploitation and exploration, which causes the population to converge too early and thus fails to further explore better solutions. Another distinct problem found in DE is known as stagnation, which was first reported in [17] and further discussed in [14]. Unlike premature convergence, the population maintains a certain level of diversity, but the algorithm still fails to find any better solutions. Mathematically, two measures are defined as follows:

$$d_G = \frac{1}{NP}\sum_{i=1}^{NP} \|\vec{x}_{i,G} - \bar{x}_G\| \tag{10}$$

where $\vec{x}_{i,G} \in \mathbf{P}_G$, $\bar{x}_G = (1/NP)\sum_{i=1}^{NP} \vec{x}_{i,G}$ is the centroid of $\mathbf{P}_G$ and

$$UN\_UP_{i,G+1} = \begin{cases} 0 & \text{if } f(\vec{u}_{i,G}) \le f(\vec{x}_{i,G}) \\ UN\_UP_{i,G} + 1 & \text{otherwise} \end{cases} \quad \text{for } i = 1, 2, \ldots, NP \tag{11}$$

In (10), d_G represents the average distance of the population $\mathbf{P}_G$ in generation G to the centroid $\bar{x}_G$, while in (11), *UN_UP* records the consecutive unsuccessful updates for each $\vec{x}_{i,G} \in \mathbf{P}_G$. Therefore, if d_G does not decrease to a small value, it indicates that $\mathbf{P}_G$ has not converged [14]. If $UN\_UP_{i,G} > T$, where T is the threshold of stagnation, $\vec{x}_{i,G}$ becomes stagnant [14].

2.4. Collective intelligence

Collective intelligence (CI) [11,12,18] is group intelligence that emerges from the collaboration, collective efforts, and competition of many individuals and appears in consensus decision-making. George Pór [27] defined the collective intelligence phenomenon as "the capacity of human communities to evolve towards higher order complexity and harmony, through such innovation mechanisms as differentiation and integration, competition and collaboration." An antecedent of the concept comes from entomologist William Morton Wheeler's finding that some animals, such as ants, can cooperate closely and become indistinguishable from a single organism. In the book *The Wisdom of Crowds* [33], published in 2004, James Surowiecki stated that the collective intelligence of a group is remarkable in some cases and often exceeds the knowledge of the smartest people within the group. Even if the majority of the group members are not experts, they can still achieve a collective wisdom [33]. In the book, the authors concentrated on three types of crowd wisdom: cognition, coordination and cooperation problems. There are four elements required to form a wise crowd: diversity of opinion (an individual in the cloud should have his/her own private information), independence (each person's opinion in the cloud is not influenced by that of the others), decentralization (people can specialize and own local knowledge) and aggregation (there is some mechanism for turning private predictions into a collective decision). It was also noted in [33] that CI may fail when the group lacks sufficient experts or when the members fail to think differently. Successful applications of CI can be found in Web 2.0 [22], Google's PageRank citation ranking [23], Wikipedia [1], online games [30], etc. In this paper, the concept of CI is introduced to DE, which treats the individual with smaller fitness as an expert and makes the population work in a collaborative manner.

3. Differential evolution powered by collective information

3.1. Motivation

As described in Section 2.2, the vectors involved in most mutation strategies are either randomly selected or the best in the population. However, multiple high-performing vectors are considered in some approaches, such as JADE [48], in which (7) is proposed, and they still rely on a single good solution.

Here, we explore the use of collective information from top performers in the population to improve DE. To express the design motivation, a simple illustrative example is given, as follows:

Consider a simple optimization problem with two variables, where the target is to minimize the following function, whose landscape is plotted in Fig. 1.

$$f(x) = (x_1 - 1)^2 + (x_2 - 1)^2, \qquad x_i \in [-5, 5] \tag{12}$$

Suppose that in a population of ten individuals, there are two solutions: A (5, 5) and B (−1, −1), with fitness rankings $\text{rank}_A = 7$ and $\text{rank}_B = 3$, respectively. To evolve solution A, the normalized collective information from solution A and various

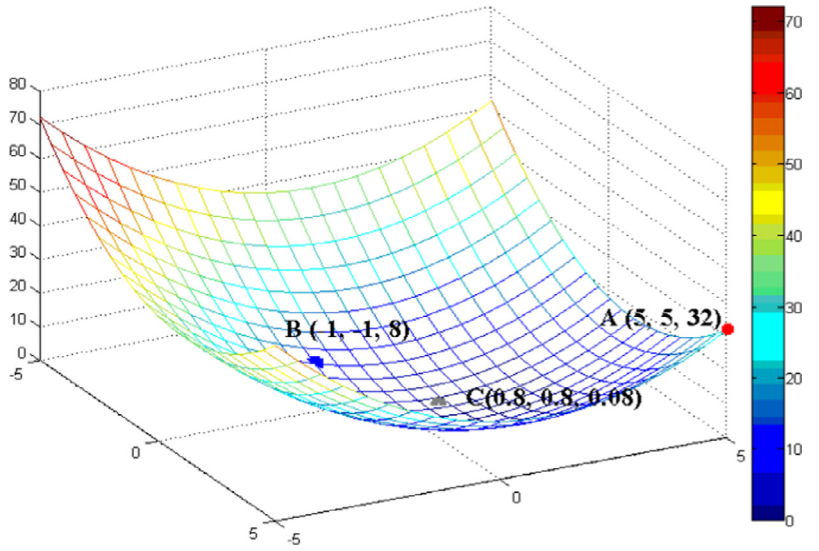

Fig. 1. Illustration of the efficiency of the collective information by a simple example.

superior solutions (e.g., solution B) with respect to A can be simply calculated as

$$C = \frac{rank_B}{rank_A + rank_B} \times A + \frac{rank_A}{rank_A + rank_B} \times B = (0.8, 0.8) \tag{13}$$

Herein, $f(C) = 0.08$ is smaller than $f(A) = 32$, indicating that solution C is superior to solution A. Moreover, C is even superior to B because $f(C) < f(B) = 8$. Therefore, the collective information C provides a very promising evolution direction and should contribute to the enhancement of the search efficiency.

Alternately, note that for more complex function landscapes, such as the composition functions, it is also possible that $f(C) > f(A)$. This, in turn, may promote diversity and benefit the exploration capability.

To take advantage of the collective information, we propose collective information-based mutation (CIM) and collective information-based crossover (CIX) operators; furthermore, we formulate a collective information-powered DE variant called CIPDE in the following sections.

3.2. Design of CIM

A new CIM mutation strategy, referred to as "current-to-ci_*m*best/1," is proposed. It utilizes the collective information of several good vectors with respect to the target vector and is described by:

$$\vec{v}_{i,G} = \vec{x}_{i,G} + F \cdot (\vec{x}_{ci\_mbest^i,G} - \vec{x}_{i,G}) + F \cdot (\vec{x}_{r_1^i,G} - \vec{x}_{r_2^i,G}) \tag{14}$$

where $\vec{x}_{i,G}$ is the target vector, with a fitness ranking of i, $\vec{x}_{r_1^i,G}$ and $\vec{x}_{r_2^i,G}$ are two randomly selected distinct ($r_1 \neq r_2 \neq i$) vectors from the current population P_G, and $\vec{x}_{ci\_mbest^i,G}$ is a collective vector computed by:

$$\vec{x}_{ci\_mbest^i,G} = \sum_{k=1}^{m} w_k \cdot \vec{x}_{k,G} \tag{15}$$

The collective vector in (15) is a linear combination of the top ranking m (m is a randomly generated integer for $\vec{x}_{i,G}$ and $m \in [1, i]$) population vectors in P_G with fitness values better than or equal to $\vec{x}_{i,G}$. The weighting factors w_k determine the contributions of different population vectors. Because a vector with a higher rank should contribute more, we let

$$w_k = \frac{m - k + 1}{1 + 2 + \ldots + m} \quad \text{for} \quad k = 1, 2, \ldots, m. \tag{16}$$

Remark 1. In CIM, m is a randomly generated integer. As later confirmed by the analyses given in Sections 4.1 and 4.5, it results in a balanced exploitative and explorative search, reflected by its outperformance compared to other strategies, such as "current-to-best/1" [32], "current-to-pbest/1" [48] and "current-to-gr_best/1" [16].

Remark 2. Because $\vec{x}_{ci\_mbest^i,G}$ is a normalized combination of the m best vectors, which are better than or equal to $\vec{x}_{i,G}$, it is expected that the resultant vector $\vec{v}_{i,G}$ would be a promising vector beneficial to the evolution of $\vec{x}_{i,G}$. Further detailed analysis of the performance of CIM will be given in Section 4.

Algorithm 2
Classic DE with the CIM operator (**CIMDE**).

1: Set the population size *NP*, mutation factor *F*, and crossover factor *CR*; initialize and evaluate the population $\mathbf{P_G}$; set the generation counter $G=0$;
2: **While** the stopping criteria are not met, **Do**
3: *Re*-index the current population $\mathbf{P_G}$ in ascending order according to the fitness;
4: **For** $i=1$: *NP* **Do**
———— CIM operation————
5: Randomly choose $\vec{x}_{r_1^i,G}$ and $\vec{x}_{r_2^i,G}$ from the current population $\mathbf{P_G}$, where $r_1 \neq r_2 \neq i$; generate $m=$ randint [1, i]; generate a collective vector $\vec{x}_{ci\_mbest^i,G}$ using Eq. (15); generate a mutant vector $\vec{v}_{i,G}$ using Eq. (14); ⇐
———— Classic crossover operation————
6: Generate $j_{\text{rand}}=$ randint[1, D];
7: **For** $j=1$: D **Do**
8: **If** rand$_j(0,1) \leq CR$ or $j=j_{\text{rand}}$, **then** $u_{i,j,G}=v_{i,j,G}$;
9: **Else** $u_{i,j,G}=x_{i,j,G}$;
10: **End If**
11: **End For**
————Selection operation————
12: Evaluate the fitness of $\vec{u}_{i,G}$;
13: **If** $f(\vec{u}_{i,G}) \leq f(\vec{x}_{i,G})$, **then** $\vec{x}_{i,G+1}=\vec{u}_{i,G}$;
14: **Else** $\vec{x}_{i,G+1}=\vec{x}_{i,G}$;
15: **End If**
16: **End For**
17: $G=G+1$;
18: **End While**

The pseudocode of classic DE with the CIM operator (CIMDE) is presented in Algorithm 2. For ease of understanding, the differences between Algorithm 2 and Algorithm 1 are highlighted using "⇐". It can be observed that line 5 in the pseudocode of Algorithm 2 is the proposed CIM strategy, which replaces the classic "rand/1" strategy used in Algorithm 1.

3.3. Design of CIX

As discussed in the literature [14,17,46], DE suffers from the problem of stagnation, which greatly affects its performance. During the optimum seeking process, some target vectors in the current population may stop updating, i.e., their unsuccessful update index *UN_UP* continually increases. To address this problem, a collective information-based crossover (CIX) is applied to the mutant vector $\vec{v}_{i,G}$ (see (14)) and the collective vector $\vec{x}_{ci\_mbest^i,G}$ (see (15)). When the ith target vector stagnates, i.e., $UN\_UP(i) > T$, where T is a threshold, CIX is to be performed as follows:

$$u_{i,j,G} = \begin{cases} v_{i,j,G} & \text{if } rand_j(0,1) \leq CR \text{ or } j = j_{rand} \\ x_{ci\_mbest^i,j,G} & \text{otherwise} \end{cases} \tag{17}$$

where $x_{ci\_mbest^i,j,G}$ is the jth dimension of $\vec{x}_{ci\_mbest^i,G}$.

The effectiveness of CIX in dealing with stagnation is illustrated by a simple 2-*D* example, as depicted in Fig. 2. Vectors 1–5 are the five population vectors with fitness ranking from 1 to 5, indicated by the blue circles in Fig. 2. $\vec{x}_{i,G}$ is a vector suffering from stagnation and is inferior to Vectors 1–5 in terms of fitness. Referring to 14)–(15) and assuming that $m=5$, a collective vector $\vec{x}_{ci\_mbest^i,G}$ and a mutant vector $\vec{v}_{i,G}$ can be obtained. Subsequently, CIX is conducted between $\vec{v}_{i,G}$ and $\vec{x}_{ci\_mbest^i,G}$ according to ((17) and the resultant trial vector $\vec{u}_{i,G}$ would be located at position I or II. In contrast, if the classical crossover operation (5) is performed, $\vec{u}_{i,G}$ would be located at position III or IV. Clearly, solutions at positions I and II are closer to $\vec{x}_{ci\_mbest^i,G}$ and satisfy $f(\vec{u}_{i,G}) < f(\vec{x}_{i,G})$, which is the condition for survival in the next generation.

In Section 4, numerical analysis will be presented to provide a comprehensive study of the effectiveness of CIX.

For ease of reference, the DE incorporating CIM and CIX is named CIMXDE; its operational procedures are listed below:

Step 1: Set the population size *NP*, mutation factor *F* and crossover factor *CR*; initialize and evaluate the population $\mathbf{P_G}$; set the generation counter $G=0$; set the stagnation tolerance *T* and initialize *UN_UP*.
Step 2: *Re*-index the current population $\mathbf{P_G}$ in ascending order according to the fitness and re-index the associated *UN_UP*.
Step 3: Perform CIM: for each target vector $\vec{x}_{i,G}$, generate a mutant vector $\vec{v}_{i,G}$ by using (14).
Step 4: Perform classic crossover or CIX: for each target vector $\vec{x}_{i,G}$, if $UN\_UP(i) > T$, perform CIX between $\vec{v}_{i,G}$ and $\vec{x}_{ci\_mbest^i,G}$ to generate a trial vector $\vec{u}_{i,G}$ using (17); otherwise, perform the classic crossover (5) between $\vec{v}_{i,G}$ and $\vec{x}_{i,G}$ to generate a trial vector $\vec{u}_{i,G}$.
Step 5: Perform selection: if $\vec{x}_{i,G}$ is better than $\vec{u}_{i,G}$ in terms of fitness, $\vec{x}_{i,G+1}=\vec{x}_{i,G}$ and increase the $UN\_UP(i)$ of $\vec{x}_{i,G}$; otherwise, $\vec{x}_{i,G+1}=\vec{u}_{i,G}$ and set $UN\_UP(i)=0$.
Step 6: If the termination criteria are met, stop the algorithm; otherwise, $G=G+1$ and go to Step 2.

The pseudocode of CIMXDE is given in Algorithm 3. The differences between Algorithm 3 and Algorithm 2 are indicated using "⇐". Lines 14–18 in Algorithm 3 describe the CIX operation, which is performed when the unsuccessful update value $UN\_UP(i)$ of individual $\vec{x}_{i,G}$ is larger than the stagnation tolerance *T*.

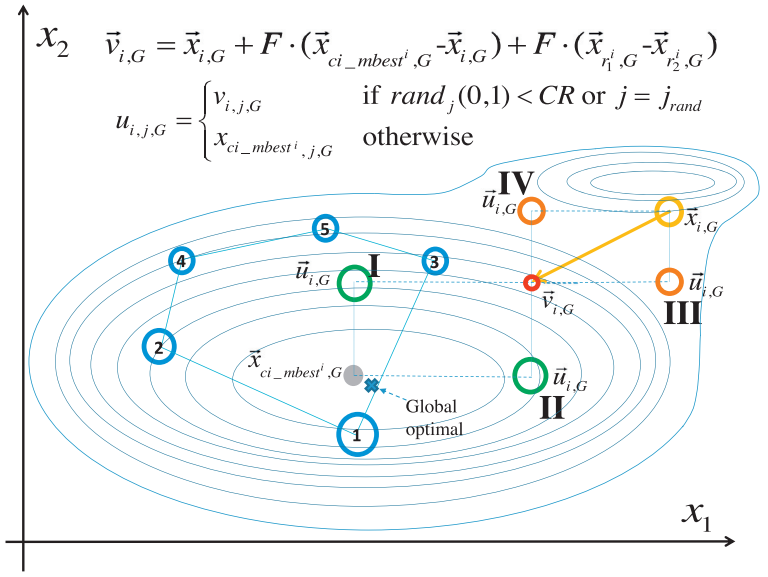

Fig. 2. Illustration of the effectiveness of CIX in handling stagnation. The contour lines indicate the corresponding objective function values.

Algorithm 3
CIMDE incorporated with CIX (**CIMXDE**).

1: Set the population size *NP*, mutation factor *F*, and crossover factor *CR*; initialize and evaluate the population $\mathbf{P_G}$; set the generation counter $G = 0$;
set the stagnation tolerance *T*; initialize *UN_UP* $(i) = 0$ (for $i = 1, 2, \ldots, NP$); ⇐
2: **While** the specific terminal conditions are not satisfied, **Do**
3: *Re*-index the current population $\mathbf{P_G}$ in ascending order according to the fitness and re-index the associated *UN_UP*; ⇐
4: **For** $i = 1$: *NP* **Do**
————————CIM operation————————-
5: Randomly choose $\vec{x}_{r_1^i,G}$ and $\vec{x}_{r_2^i,G}$ from the current population, where $r_1 \neq r_2 \neq i$; generate $m =$ randint [1, *i*]; generate a collective vector $\vec{x}_{ci\_mbest^i,G}$ using Eq. (15); generate a mutant vector $\vec{v}_{i,G}$ using Eq. (14);
6: **If** *UN_UP* $(i) \leq T$
————————-Classic crossover operation————————-
7: Generate $j_{\text{rand}} = \text{randint}[1, D]$;
8: **For** $j = 1$: *D* **Do**
9: **If** $\text{rand}_j(0,1) \leq CR$ or $j = j_{\text{rand}}$, **then** $u_{i,j,G} = v_{i,j,G}$;
10: **Else** $u_{i,j,G} = x_{i,j,G}$;
11: **End If**
12: **End For**
13: **Else** ⇐
————————CIX operation————————
14: **For** $j = 1$: *D* **Do** ⇐
15: **If** $\text{rand}_j(0,1) \leq CR$ or $j = j_{\text{rand}}$, **then** $u_{i,j,G} = v_{i,j,G}$; ⇐
16: **Else** $u_{i,j,G} = x_{ci\_mbest^i,j,G}$; ⇐
17: **End If** ⇐
18: **End For** ⇐
19: **End If**
————————-Selection operation ————————
20: Evaluate the fitness of $\vec{u}_{i,G}$;
21: **If** $f(\vec{u}_{i,G}) \leq f(\vec{x}_{i,G})$, **Then**
22: $\vec{x}_{i,G+1} = \vec{u}_{i,G}$;
23: *UN_UP* $(i) = 0$; ⇐
24: **Else**
25: $\vec{x}_{i,G+1} = \vec{x}_{i,G}$;
26: *UN_UP* $(i) =$ *UN_UP* $(i) + 1$; ⇐
27: **End If**
28: **End For**
29: $G = G + 1$;
30: **End While**

3.4. Parameter adaptation

To further enhance the performance, parameter adaptation schemes are incorporated into CIM and CIX. Various possible approaches [3,36,48] have been evaluated, and experimental results have shown that the one in JADE [48] is the most effective. Its procedures are described in the following.

For each individual $\vec{x}_{i,G}$ in generation G, the parameter F_i is generated according to a Cauchy distribution with location parameter μ_F and scale parameter 0.1.

$$F_i = \text{rand}_{Ci}(\mu_F, 0.1) \tag{18}$$

If $F_i > 1.0$, it is truncated to 1.0; if $F_i < 0$, it is regenerated using (18). μ_F is initialized to 0.7 and updated according to (19) in each generation.

$$\mu_F = (1 - c) \cdot \mu_F + c \cdot mean_L(S_F) \tag{19}$$

where c is a positive constant within [0, 1], S_F is the set of successful F values and $mean_L(\cdot)$ is the Lehmer mean. The Lehmer mean is used instead of the arithmetic mean to propagate larger F values for the improvement of the population diversity [48].

$$mean_L = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F} \tag{20}$$

Similarly, for each individual $\vec{x}_{i,G}$ at generation G, its corresponding CR is generated by a normal distribution with mean μ_{CR} and standard deviation 0.1.

$$CR_i = \text{randn}_i(\mu_{CR}, 0.1) \tag{21}$$

CR_i will be regenerated if $CR_i > 1$ or $CR_i < 0$; μ_{CR} is initialized to 0.5 and updated in each generation as follows:

$$\mu_{CR} = (1 - c) \cdot \mu_{CR} + c \cdot mean_A(S_{CR}) \tag{22}$$

where c is a positive constant within [0, 1], S_{CR} is the set of successful CR values, and $mean_A(\cdot)$ is the arithmetic mean.

3.5. Design of CIPDE

By combining CIM, CIX and the parameter adaptation method in Section 3.4, a new DE variant, referred to as CIPDE, is formulated; its pseudocode is given in Algorithm 4. The arrows "⇐" highlight the differences between Algorithm 4 and Algorithm 3, i.e., the parameter adaptation.

As seen from Algorithm 4, CIPDE features the collective information of the population. For each target individual $\vec{x}_{i,G}$ in the population, a collective information-based vector $\vec{x}_{ci\_mbest^i,G}$ is generated for use in the CIM mutation operation (Line 7). Moreover, if $\vec{x}_{i,G}$ falls into stagnation, its corresponding $\vec{x}_{ci\_mbest^i,G}$ is further used in the CIX crossover operation to help the algorithm escape from the situation of stagnation (Lines 16–20). The superiority of collective information-based mutation and crossover operations over those without collective information utilization will be comprehensively investigated in Section 4.

4. Experimental study

In this section, the effectiveness of the proposed CIM and CIX operators and the superiority of the presented CIPDE algorithm are verified by comprehensive experiments conducted on the CEC2013 benchmark function set [19]. The CEC2013 set includes a wide range of optimization functions, such as unimodal functions (F1−F5), basic multimodal functions (F6−F20) and composition functions (F21−F28).

The performance of different algorithms is evaluated based on the solution error value, which is defined as $f(x) - f(x^*)$, where $f(x^*)$ is the global optimal solution of the test function and $f(x)$ is the smallest error value obtained after $10^4 \times D$ function evaluations (FES) [19]. The compared algorithms were run independently on every function 51 times. The mean and standard deviation of the solution error values are reported. As suggested in [19], the solution error values smaller than 10^{-8} are treated as zero. Moreover, to have a statistically sound conclusion, error values achieved by different algorithms are compared according to the Wilcoxon signed-rank test with a significance level of 0.05. The symbols "-," "=" and "+" represent that the performance of the compared algorithms is significantly worse than, similar to or better than that of the considered algorithm, respectively. The smallest mean error values achieved on each function are highlighted in **bold**.

4.1. Comparison of CIM with other competitive mutation operators

Firstly, the proposed "current-to-ci_*m*best/1" (CIM) operator is compared with three other competitive mutation strategies in the literature: "current-to-best/1" [32], "current-to-pbest/1" [48] and "current-to-gr_best/1" [16], which are also described in Eqs. (3), (7), (8), respectively. Parameter settings for the compared strategies are summarized in Table 1.

Algorithm 4
CIMXDE with parameter adaptation (**CIPDE**).

1: Set the population size NP; initialize and evaluate the population $\mathbf{P_G}$; set the generation counter $G = 0$; set the stagnation tolerance T; initialize $UN\_UP\ (i) = 0$ (for $i = 1, 2, \ldots, NP$); set the location parameter μ_F, mean parameter μ_{CR} and parameter c; ⇐
2: **While** the specific terminal conditions are not satisfied, **Do**
3: $S_F = \emptyset, S_{CR} = \emptyset$; ⇐
4: *Re*-index the current population $\mathbf{P_G}$ in ascending order according to the fitness and re-index the associated UN_UP;
5: **For** $i = 1$: NP **Do**
6: Generate $F_i = \text{rand}_{Ci}\ (\mu_F, 0.1)$, $CR_i = \text{randn}_i\ (\mu_{CR}, 0.1)$; ⇐
————CIM operation————-
7: Randomly choose $\vec{x}_{r_1^i,G}$ and $\vec{x}_{r_2^i,G}$ from the current population, where $r_1 \neq r_2 \neq i$; generate $m = \text{randint}[1, i]$; generate the linear combination vector $\vec{x}_{ci\_mbest^i,G}$ using Eq. (15); generate a mutant vector $\vec{v}_{i,G}$ using Eq. (14);
8: **If** $UN\_UP\ (i) \leq T$
———Classic crossover operation—————
9: Generate $j_{\text{rand}} = \text{randint}[1, D]$;
10: **For** $j = 1$: D **Do**
11: **If** $\text{rand}_j(0,1) \leq CR_i$ or $j = j_{\text{rand}}$, **then** $u_{i,j,G} = v_{i,j,G}$;
12: **Else** $u_{i,j,G} = x_{i,j,G}$;
13: **End If**
14: **End For**
15: **Else**
———CIX operation———
16: **For** $j = 1$: D **Do**
17: **If** $\text{rand}_j(0,1) \leq CR_i$ or $j = j_{\text{rand}}$, **then** $u_{i,j,G} = v_{i,j,G;}$
18: **Else** $u_{i,j,G} = x_{ci\_mbest^{i},j,G}$;
19: **End If**
20: **End For**
21: **End If**
———-Selection operation ———
22: Evaluate the fitness of $\vec{u}_{i,G}$;
23: **If** $f(\vec{u}_{i,G}) \leq f(\vec{x}_{i,G})$, **Then**
24: $\vec{x}_{i,G+1} = \vec{u}_{i,G}$;
25: $UN\_UP\ (i) = 0$;
26: $F_i \rightarrow S_F$; $CR_i \rightarrow S_{CR}$; ⇐
27: **Else**
28: $\vec{x}_{i,G+1} = \vec{x}_{i,G}$;
29: $UN\_UP\ (i) = UN\_UP\ (i) + 1$;
30: **End If**
31: **End For**
———Parameter adaptation———
32: $\mu_F = (1 - c) \cdot \mu_F + c \cdot mean_L(S_F)$; ⇐
33: $\mu_{CR} = (1 - c) \cdot \mu_{CR} + c \cdot mean_A(S_{CR})$; ⇐
34: $G = G + 1$;
35: **End While**

Table 1
Parameter settings for the four compared mutation strategies.

Mutation strategy	Parameter settings
Current-to-best/1	$F = 0.7$, $CR = 0.5$, $NP = 100$
Current-to-pbest/1	$F = 0.7$, $CR = 0.5$, $NP = 100$, $p = 0.05$ [48]
Current-to-gr_best/1	$F = 0.7$, $CR = 0.5$, $NP = 100$, $q = 0.15$ [16]
CIM	$F = 0.7$, $CR = 0.5$, $NP = 100$

Table 2
Comparison results of the three mutation strategies with CIMDE according to the Wilcoxon signed-rank test with a significance level of 0.05.

$D = 30$ Algorithm	−/=/+	$D = 50$ Algorithm	−/=/+
DE/current-to-best/1/bin	17/8/3	DE/current-to-best/1/bin	21/6/1
DE/current-to-pbest/1/bin	20/4/4	DE/current-to-pbest/1/bin	19/7/2
DE/current-to-gr_best/1/bin	20/4/4	DE/current-to-gr_best/1/bin	19/6/3

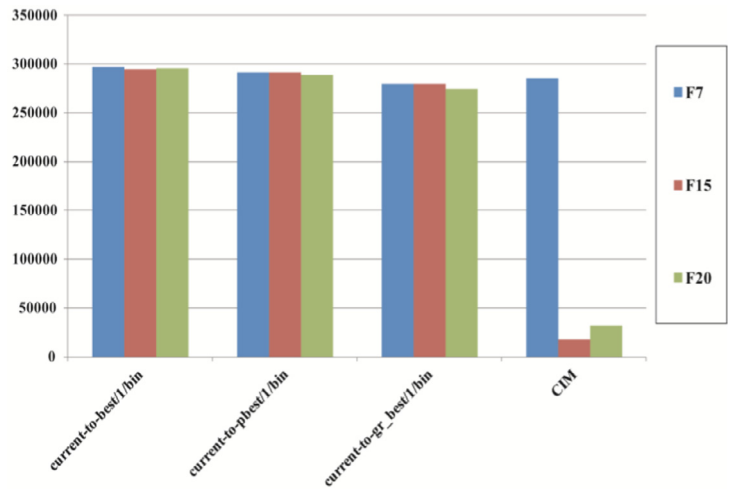

Fig. 3. The mean values of *Num_sup_i* obtained by the compared mutation strategies on the 30-dimensional functions F7, F15 and F20 over 51 runs.

The mean and standard deviation of error values achieved on 30- and 50-dimensional CEC2013 functions with 51 independent runs are shown in Tables S1 and S2 in the supplementary file, respectively. The comparison results are summarized in Table 2.

As shown in Tables S1, S2 and 2, CIM performs significantly better than the other three strategies for both 30- and 50-dimensional cases. To be specific, in terms of 30-dimensional functions, CIM significantly outperforms "current-to-best/1," "current-to-pbest/1" and "current-to-gr_best/1" in 17, 20 and 20 cases but only underperforms in 3, 4 and 4 cases out of the total 28 functions, respectively. In terms of 50-dimensional functions, CIM wins in 21, 19 and 19 cases and loses in 1, 2 and 3 cases, respectively.

According to the mathematical characteristics of the test functions, CIM can perform well on different types of functions, including unimodal (F1-F5), multimodal (F6-F20) and composition (F21-F28) functions.

Specifically, we evaluate three functions, i.e., F7, F15, and F20, to study the exploration and exploitation capabilities of the algorithms. The rotated Schaffer's F7 function (F7) and the rotated Schwefel's function (F15) both have a huge number of local optima, imposing great difficulty in optimization [36]. They can be used to examine the exploration ability of an algorithm. Meanwhile, the expanded Schaffer's F6 function (F20) is very suitable for investigating the exploitation capability [36]. Fig. S1 in the supplementary file presents the evolution process of median error values versus function evaluations (FES) obtained by the compared mutation strategies on these three functions. As shown in Fig. S1, together with Table S1, CIM obtains significantly smaller error values on both F7 and F15. This indicates the superiority of CIM in facilitating escape from local optima compared with other strategies. Alternately, with respect to function F20, CIM also significantly outperforms the others, showing that CIM also maintains a very good exploitative capability. As a remark, CIM is very competitive thanks to the collective vector $\vec{x}_{ci\_mbest^i,G}$ provided by promising solutions.

To further investigate the search mechanism of CIM, another experiment is conducted. The terminal vector of the first difference vector (i.e., $\vec{x}_{best,G}$, $\vec{x}_{pbest^i,G}$, $\vec{x}_{gr\_best^i,G}$ and $\vec{x}_{ci\_mbest^i,G}$) in the four compared strategies is compared with its corresponding target vector $\vec{x}_{i,G}$ and the best vector $\vec{x}_{best,G}$ in the current population, respectively. The variables *Num_sup_i* and *Num_sup_best* denote the total number of times that the four terminal vectors are superior to $\vec{x}_{i,G}$ and $\vec{x}_{best,G}$ in terms of fitness values out of the total $10^4 \times D$ FES, respectively. The mean values of *Num_sup_i* and *Num_sup_best* achieved on F7, F15 and F20 over 51 independent runs are depicted in Figs. 3 and 4, respectively.

It can be observed from Fig. 3 that the *Num_sup_i* values of the other three competitors (i.e., "current-to-best/1," "current-to-pbest/1" and "current-to-gr_best/1") are almost equal to the maximum FES, i.e., 300,000. As for CIM, the *Num_sup_i* value on F7 is similar to that of other strategies. However, on F15 and F20, *Num_sup_i* is significantly smaller, which means that CIM is more likely to explore more search space rather than the restricted areas with better solutions by other strategies.

From Fig. 4, *Num_sup_best* of $\vec{x}_{ci\_mbest^i,G}$ is clearly larger than those of other competitors, which have *Num_sup_best* = 0. Among the three considered functions, the value of *Num_sup_best* is largest for F7, followed by F20 and F15. This means that the use of $\vec{x}_{ci\_mbest^i,G}$ in CIM can provide potential solutions that are even better than the current best vector $\vec{x}_{best,G}$. Thus, it leads to more potential promising areas, making CIM exploitative and efficient.

From Table S1 and Figs. 3 and 4, it can be concluded that CIM exhibits a more efficient search ability than the other three mutation strategies.

4.2. Effectiveness of the CIX operator in dealing with stagnation

As mentioned in Section 3.3, CIX is designed to prevent the target vectors from stagnation. To investigate this effect, the performances of CIMDE and CIMXDE are compared based on the parameter settings given in Table 3.

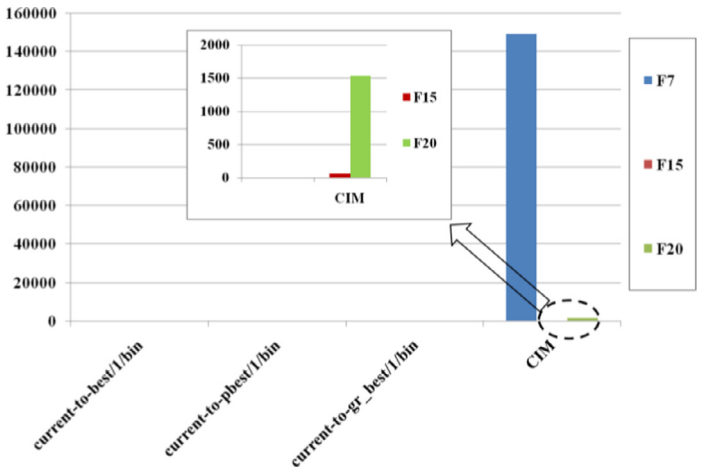

Fig. 4. The mean values of *Num_sup_best* obtained by the compared mutation strategies on the 30-dimensional functions F7, F15 and F20 over 51 runs.

Table 3
Parameter settings for CIMDE and CIMXDE.

Algorithm	Parameter settings
CIMDE	$NP = 100$, $c = 0.1$, $\mu_F = 0.7$, $\mu_{CR} = 0.5$
CIMXDE	$NP = 100$, $c = 0.1$, $\mu_F = 0.7$, $\mu_{CR} = 0.5$, $T = 90$

Table 4
Comparison results of CIMDE with CIMXDE according to the Wilcoxon signed-rank test with a significance level of 0.05.

D	$-/=/+$
30	16/11/1
50	15/10/3

The mean and standard deviation of solution error values obtained by these two algorithms are presented in Table S3 in the supplementary file, while Table 4 summarizes the comparison results. As observed, the CIMDE without CIX performs significantly worse than CIMXDE on both 30- and 50-dimensional functions. Out of the total 56 cases, CIMDE loses in 31 $(= 16 + 15)$ functions and only wins in 4 $(= 1 + 3)$ functions.

Fig. 5 depicts the average distance d_G of all individuals to the centroid of the population achieved by CIMDE and CIMXDE versus function evaluations (FES) on the 30-dimensional unimodal function F2 and multimodal function F9, respectively. As shown, CIMXDE obtains a much smaller d_G on F2 and F9 than CIMDE. This indicates that CIX can speed up the convergence of CIMDE and help the algorithm escape from the first phenomenon of stagnation. The average consecutive unsuccessful update indices $\overline{UN\_UP}$ of all individuals achieved by CIMDE and CIMXDE versus function evaluations (FES) on the 30-dimensional unimodal function F2 and multimodal function F9, respectively, are also plotted in Fig. 6. CIMXDE maintains much smaller $\overline{UN\_UP}$ values, implying that CIX can also handle the second phenomenon of stagnation. This confirms the illustration depicted previously in Fig. 2 that trial vectors created by CIX have a higher chance of surviving than those created by the classic crossover operation.

4.3. Comparison of CIPDE with state-of-the-art DE variants

In this subsection, the proposed CIPDE algorithm is compared with seven state-of-the-art DE variants: JADE [48], SaDE [28], EPSDE [20], jDE [3], CoDE [38], SHADE [35] and CoBiDE [40].

These DE variants are selected due to their well-known competitive performance and high popularity. Moreover, they represent different types of improvements that were discussed in Section 2.2.

- JADE and jDE are DEs with adaptive control parameters.
- SaDE and EPSDE are DEs with adaptive mutation strategies and control parameters.
- CoDE is a composite DE with multiple mutation strategies and control parameters.
- SHADE is an enhanced JADE variant with an improved “current-to-pbest/1” strategy and a success history-based parameter adaptation scheme.
- CoBiDE is an advanced DE with covariance matrix learning crossover and control parameter adaptation.

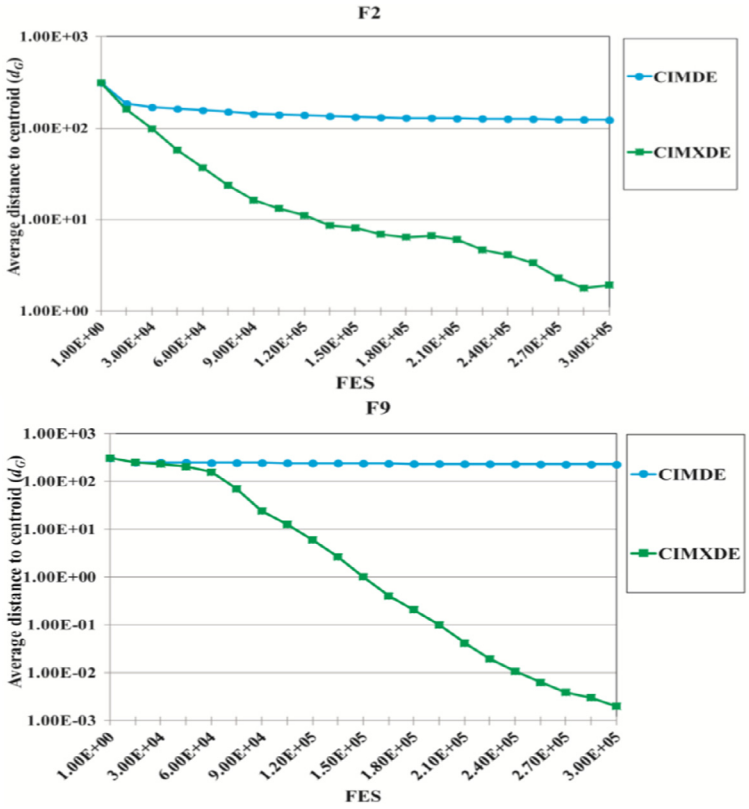

Fig. 5. Evolution of the average distance to the centroid of the population on the 30-dimensional functions F2 and F9.

Table 5
Comparison results of CIPDE with seven state-of-the-art DE variants according to the Wilcoxon signed-rank test with a significance level of 0.05.

$D = 30$		$D = 50$		$D = 100$	
Algorithm	−/=/+	Algorithm	−/=/+	Algorithm	−/=/+
JADE	12/10/6	JADE	11/11/6	JADE	12/9/7
SaDE	22/4/2	SaDE	20/6/2	SaDE	20/5/3
EPSDE	19/7/2	EPSDE	21/5/2	EPSDE	21/5/2
jDE	13/11/4	jDE	11/8/9	jDE	14/4/10
CoDE	17/5/6	CoDE	17/3/8	CoDE	18/3/7
SHADE	13/8/7	SHADE	13/7/8	SHADE	10/8/10
CoBiDE	16/8/4	CoBiDE	14/9/5	CoBiDE	15/4/9

The parameter settings for the compared DEs are the same as those recommended in the original literature, while the parameter settings for CIPDE are: $NP = 100$, $c = 0.1$, $\mu_F = 0.7$, $\mu_{CR} = 0.5$, and $T = 90$.

The mean and standard deviation of error values achieved by these algorithms on the 30-dimensional CEC2013 benchmark functions are presented in Table S4 in the supplementary file. The comparison results given by the Wilcoxon signed-rank test at a significance level of 0.05 are summarized in Table 5.

As shown, CIPDE exhibits significantly better performance compared to the other DE variants at $D = 30$. To be specific, CIPDE outperforms JADE, SaDE, EPSDE, jDE, CoDE, SHADE and CoBiDE in 12, 22, 19, 13, 17, 13 and 16 cases and only underperforms in 6, 2, 2, 4, 6, 7 and 4 cases out of the 28 total cases, respectively. With respect to the characteristics of the test functions, the following results can be observed:

For the unimodal functions F1-F5, JADE and SHADE perform the best, followed by CIPDE. CIPDE loses in 2 functions (F3 and F4) and ties in 3 functions when compared to JADE and SHADE, respectively. Nevertheless, compared with the other five DE variants, CIPDE exhibits either better or similar performance.

For the basic multimodal functions F6-F20, CIPDE achieves the best performance. Specifically, CIPDE outperforms JADE, SaDE, EPSDE, jDE, CoDE, SHADE and CoBiDE in 8 (F7, F9, F12, F13, F15 and F18-F20), 12 (F6, F7, F10-F15 and F17-F20), 10 (F6, F7, F9, F10, F12, F13, F15 and F18-F20), 8 (F6, F9, F12, F13, F15 and F18-F20), 9 (F6, F7, F12-F15 and F18-F20), 8 (F9, F10, F12, F13, F15 and F18-F20) and 9 (F6, F12-F15 and F17-F20) cases, respectively. Compared to the same set of competitors, CIPDE

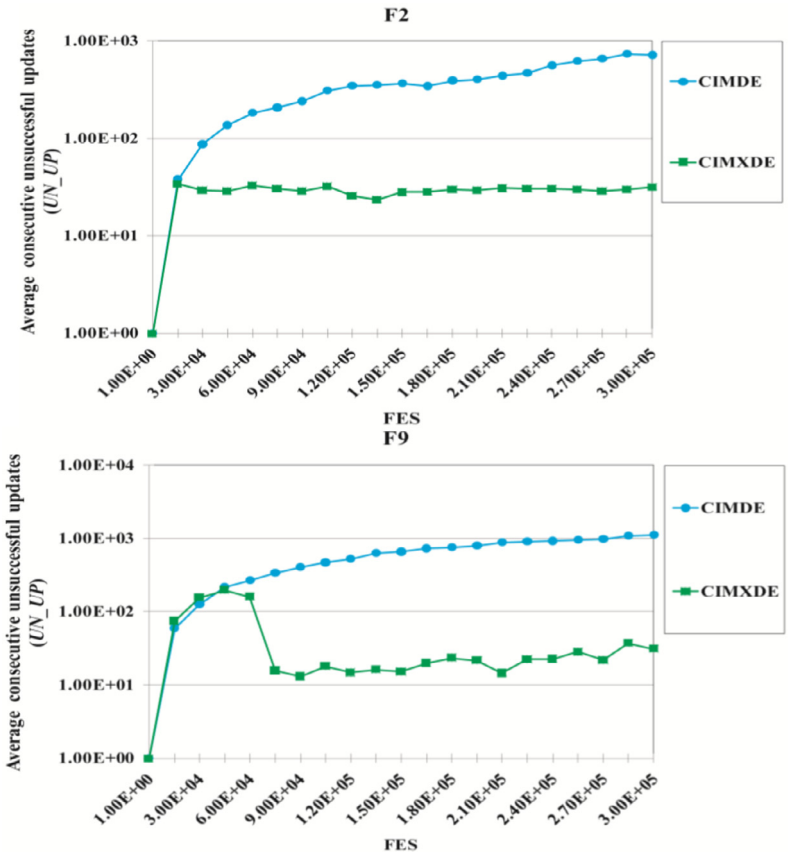

Fig. 6. Evolution of the average consecutive unsuccessful update number on the 30-dimensional functions F2 and F9.

underperforms in 3 (F14, F16 and F17), 1 (F9), 2 (F14 and F17), 3 (F10, F14 and F17), 5 (F8-F10 and F16-F17), 4 (F8, F14, F16 and F17) and 3 (F9, F10 and F16) cases, respectively. CIPDE achieves much smaller error values than the competitors on F12, F13, F15 and F18-F20.

Regarding the composition functions F21-F28 with complex mathematical characteristics that are very difficult to optimize, Table S4 shows that CIPDE also performs the best. CIPDE beats JADE, SaDE, EPSDE, jDE, CoDE, SHADE and CoBiDE in 4 (F23-F25 and F27), 7 (F21-F24 and F26-F28), 6 (F22-F27), 2 (F23 and F26), 5 (F22-F24, F26 and F27), 5 (F23 and F25-F28) and 6 (F21-F23 and F26-F28) functions and loses in 1 (F22), 1 (F25), 0, 1 (F25), 1 (F25), 1 (F22) and 1 (F25) function, respectively.

Figs. S2–S4 plot the evolution of median error values obtained by the seven compared variants and CIPDE on nine selected 30-dimensional benchmark functions. It clearly illustrates the superiority of CIPDE compared with the others.

To further investigate the performance of CIPDE on higher-dimensional functions, the algorithms are also compared on 50- and 100- dimensional functions. The experimental results are given in Tables S5 and S6, respectively; the comparison results are tabulated in Table 5. They indicate that CIPDE still exhibits overall superior performance. To be specific, CIPDE is statistically better than JADE, SaDE, EPSDE, jDE, CoDE, SHADE and CoBiDE in 11, 20, 21, 11, 17, 13 and 14 functions and worse in 6, 2, 2, 9, 8, 8 and 5 functions, respectively, on the 50-dimensional version of the CEC2013 functions. For the 100-dimensional functions, CIPDE performs significantly better than the compared DE variants on 12, 20, 21, 14, 18, 10 and 15 functions, respectively.

From Tables S4–S6, note that CIPDE performs consistently better than most of its competitors on functions F12, F13, F15, F18-F20 and F23 and worse than most of its competitors on functions F10, F17 and F25 in all of the 30-, 50- and 100-dimensional cases. This is not surprising when considering the No Free Lunch (NFL) theorems [44]. Although CIPDE does not obtain the best solutions on each function, it achieves the overall best performance.

Table 6 gives the overall performance ranking of all algorithms on the 30-, 50- and 100-dimensional CEC2013 functions by Friedman's test. As shown, CIPDE achieves the smallest ranking value (3.30).

4.4. Benefit of parameter adaptation to the performance of CIPDE

It is interesting to investigate the influence of parameter adaptation on the performance of CIPDE. Thus, the comparison results between CIMXDE and CIPDE are tabulated in Table S7 in the supplementary file.

Table 6
Overall performance ranking of the compared DE variants on the 30-, 50- and 100-dimensional CEC2013 benchmark set by Friedman's test.

Algorithm	Ranking ($D = 30$, 50 and 100)
JADE	4.11
SaDE	6.18
EPSDE	6.73
jDE	4.11
CoDE	4.24
SHADE	3.51
CoBiDE	3.79
CIPDE	**3.30**

Table 7
Comparison results of CIPDE variants with CIPDE according to the Wilcoxon signed-rank test with a significance level of 0.05.

$D = 30$		$D = 50$	
Algorithm	−/=/+	Algorithm	−/=/+
Variant-I	15/9/4	Variant-I	12/11/5
Variant-II	15/9/4	Variant-II	18/7/3
Variant-III	8/16/4	Variant-III	8/14/6
Variant-IV	11/15/2	Variant-IV	8/17/3
Variant-V	14/11/3	Variant-V	14/12/2
Variant-VI	9/18/1	Variant-VI	6/22/0
Variant-VII	14/12/2	Variant-VII	10/16/2

It can be observed that CIPDE performs slightly better than CIMXDE in 30-dimensional cases and slightly underperforms in 50-dimensional cases, specified by the "−/=/+" values of "13/5/10" and "11/4/13," respectively. Considering the previously employed functions F7, F15 and F20, which are useful to reveal the exploration and exploitation abilities, it is observed that the parameter adaptation makes CIMXDE greedy. CIPDE loses to CIMXDE on F7 and F15 but wins on F20. Table S7 also shows that CIPDE performs well on unimodal and basic multimodal functions, while CIMXDE is promising in composition functions. With respect to the good performance of CIPDE shown in Section 4.3, CIMXDE is indeed also competitive.

4.5. Benefit of collective information to the performance of CIPDE

To comprehensively evaluate the contributions of collective information to the performance of the CIPDE algorithm, seven variants, denoted as Variants I-VII, are constructed; their differences with respect to CIPDE are listed as follows:

Variant-I: This is a variant based on mutation (SIM) and crossover (SIX) operations that utilizes the information from single solutions. To be specific, it replaces the $\vec{x}_{ci\_mbest^i,G}$ vector in (14) and (17) with the better vector $\vec{x}_{better^i,G}$, which is randomly selected from the current population while $f(\vec{x}_{better^i,G}) \leq f(\vec{x}_{i,G})$ is satisfied.
Variant-II: It employs a greedy scheme that replaces the $\vec{x}_{ci\_mbest^i,G}$ vector in (14) and (17) with the best vector $\vec{x}_{best,G}$ from the current population.
Variant-III: SIM and CIX are adopted for mutation and crossover operations, respectively.
Variant-IV: CIM and SIX are adopted for mutation and crossover operations, respectively.
Variant-V: Different from Variant-II, Variant-V merely replaces the $\vec{x}_{ci\_mbest^i,G}$ vector of CIM with $\vec{x}_{best,G}$ and keeps CIX unchanged.
Variant-VI: Different from Variant-V, the $\vec{x}_{ci\_mbest^i,G}$ vector of CIX is replaced by $\vec{x}_{best,G}$, while CIM is kept unchanged.
Variant-VII: This is the same as CIPDE, except that the value of m in (15) is set as $m = i$.

As a summary, Variant-I and Variant-II are introduced to verify the overall benefit of collective information provided by CIM and CIX. Variants III and V are employed to confirm the superiority of CIM over SIM, while Variants IV and VI are utilized to investigate the advantages of CIX over SIX. Lastly, Variant-VII is used to confirm the effectiveness of the use of the variable m in the design.

Parameter settings of the seven variants are the same as those of CIPDE: $NP = 100$, $c = 0.1$, $\mu_F = 0.7$, $\mu_{CR} = 0.5$, and $T = 90$. The experimental results on 30- and 50-dimensional CEC2013 functions are presented in Tables S8–S11 in the supplementary file; the comparison results are summarized in Table 7. By comparing the variants with CIPDE, it can be found that:

(1) From Tables S8, S10 and 7, it is apparent that Variants I and II, which rely on single information mutation and crossover operators, perform significantly worse than CIPDE in both 30- and 50-dimensional cases, losing in 27 ($= 15 + 12$) and 33 ($= 15 + 18$) cases and winning in 9 ($= 4 + 5$) and 7 ($= 4 + 3$) cases, respectively. With respect to

the superior performance of CIPDE over Variants I and II on F7, F15 and F20 from Tables S8 and S10, it is confirmed that the collective information enhances the exploration and exploitation capabilities of CIPDE.

(2) By comparing Variants III and IV with CIPDE, the benefits of CIM and CIX can be observed. Table 7 shows that CIPDE is superior to Variants III and IV in 16 (= 8 + 8) and 19 (= 11 + 8) and inferior in 10 (= 4 + 6) and 5 (= 2 + 3) cases, respectively. Therefore, it can be concluded that both CIM and CIX contribute to the superiority of CIPDE over Variant I.

(3) Comparing CIPDE with Variants V and VI, the advantages of collective information-based operations over the single best vector-based operations can be observed from Table 7. CIPDE outperforms Variants V and VI in 28 (= 14 + 14) and 15 (= 9 + 6) and underperforms in 5 (= 3 + 2) and 1 (= 1 + 0) cases, respectively. CIPDE exhibits both superior exploration and superior exploitation abilities over Variants V and VI. It achieves better or at least similar performance compared to the two competitors on F7, F15 and F20. This is in accord with the conclusion drawn in Section 4.1.

(4) From Table 7, Variant-VII with the setting of $m = i$ is inferior to CIPDE. It loses to CIPDE in 24 (= 14 + 10) and wins in 4 (= 2 + 2) cases. The setting of $m \in$ randint[1, i] makes CIPDE more exploitative. Another advantage is that it yields increased time-savings compared to Variant-VII, which will be investigated in Section 4.6.

4.6. Algorithm complexity

In this subsection, the time complexities of different algorithms, including the classic DE variants with different mutation strategies, CIMDE, CIMXDE, CIPDE, Variant-VII and the seven state-of-the-art DE variants, are investigated and compared. All the algorithms are implemented using MATLAB and run on an Intel Core i7 3.4-GHz PC with 4 GB of RAM in a Windows 7 environment.

Tables S12 and S13 give the T_0, T_1 and T_2 values of these fourteen algorithms based on the 30- and 50- dimensional cases in the CEC2013 test suit, respectively. T_0 is a reference as the CPU time cost to run the below test program [19]:

```
for i = 1: 1,000,000
x= 0.55 + (double) i;
x = x + x; x = x./2; x = x*x; x = sqrt(x);
x = log(x); x = exp(x); y = x/x;
end
```

T_1 is the computation time needed to evaluate F14 with 200,000 FES; T_2 is the average CPU time cost for an algorithm to solve F14 with 200,000 FES over 5 runs. (T_2-T_1)/ T_0 gives the complexity of an algorithm [19].

From Tables S12 and S13, it can be observed that (1) DE/current-to-best/1/bin has the lowest complexity with the smallest (T_2-T_1)/ T_0 value, while EPSDE has the highest complexity. (2) Compared with the other three classic DE variants, CIMDE has higher complexity. This is because CIMDE requires extra CPU time to calculate the $\vec{x}_{ci\_mbest^i,G}$ vector. (3) Compared with the state-of-the-art DE variants, the complexity of CIPDE is higher than that of JADE, SaDE and jDE but lower than that of EPSDE, CoDE, SHADE and CoBiDE. (4) Compared with Variant-VII, CIPDE yields increased time-savings.

To further study the runtime of the compared algorithms on different functions requiring different amounts of evaluation cost, the average CPU time cost on the 30-dimensional functions F2, F9 and F27 over 5 independent runs is given in Table S14. It indicates that for function F2 with low function evaluation cost, the CPU time differences between CIMDE and other classic DEs and CIPDE and other advanced DE variants are significant. However, for functions F9 and F27, whose function evaluations are much more time-consuming because the running time is mainly occupied by the function evaluations rather than the complexity of the algorithms, the overhead of computing the vector $\vec{x}_{ci\_mbest^i,G}$ is trivial.

4.7. Parameter sensitivity

4.7.1. Performance sensitivity to the stagnation threshold t

In CIPDE, the stagnation threshold T controls the execution frequency of CIX. To investigate the performance sensitivity to this parameter, four variants with different T values (10, 50, 130 and 170) are compared with the standard CIPDE, where $T = 90$. All other parameter settings remain unaltered to allow a direct comparison; the experimental results are presented in Table S15.

Based on the Wilcoxon signed-rank test shown in Table S15, it can be observed that T values that are too small or too large are not suitable for CIPDE. This is because T values that are too small force CIX to perform frequently, which in turn significantly deteriorates CIPDE's exploration ability and makes the algorithm too greedy. This can be reflected by the performance on F7, F15 and F20. Alternately, T values that are too large mean that CIX is idle and that its benefits are simply weakened. From the comparison results given by the Wilcoxon signed-rank test with a 0.05 significance level, the standard CIPDE with $T = 90$ exhibits the best overall performance.

4.7.2. Performance sensitivity to the population size NP

The impact of the population size NP on the performance of CIPDE is also investigated. Four CIPDE variants with $NP = 50$, 150, 200, and 250 are considered; their performances are compared with that of the standard CIPDE with $NP = 100$. The

experimental results on 30-dimensional CEC2013 functions are summarized in Table S16. The results show that the CIPDE variants with $NP = 50$ and $NP = 250$ perform significantly worse than the standard CIPDE. Regarding $NP = 150$ and $NP = 200$, the overall performance of these two variants is similar to that of the standard CIPDE, even though different NP values are suitable for functions with different characteristics. Considering the number of smallest error values highlighted in bold in Table S16, the case with $NP = 100$ achieves the maximum number of 9.

5. Conclusions

In this paper, a DE variant called CIPDE that is powered by collective information is proposed. A collective information-based mutation (CIM) operator and a collective information-based crossover (CIX) operator are presented. CIM creates promising potential solutions by utilizing the normalized linear combination of the m best vectors that are better than the target vectors in terms of fitness values. CIX is designed to prevent stagnation by efficiently utilizing the collective information to update and converge the population. Extensive numerical experiments have been conducted. The results confirm the superiority of CIM over other competitive mutation strategies in the literature and the effectiveness of CIX in handling stagnation. The performance of CIPDE is also compared with seven popular state-of-the-art DE variants; it is concluded that CIPDE outperforms all of the others.

Acknowledgments

The work described in this paper was supported in part by the National Natural Science Foundation of China (No. 61401523), in part by the Foundation for Distinguished Young Talents in Higher Education of Guangdong, China (No. 2014KQNCX002), in part by the International Science & Technology Cooperation Program of China (No. 2015DFR11050), and in part by the External Cooperation Program of Guangdong Province of China (No. 2013B051000060).

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ins.2017.02.055.

References

[1] O. Arazy, W. Morgan, R. Patterson, Wisdom of the crowds: decentralized knowledge construction in Wikipedia, in: Proceedings of 16th Annual Workshop on Information Technologies & Systems (WITS), 2006.
[2] Y. Bi, D. Srinivasan, X. Lu, Z. Sun, W. Zeng, Type-2 fuzzy multi-intersection traffic signal control with differential evolution optimization, Expert Syst. Appl. 41 (2014) 7338–7349.
[3] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10 (2006) 646–657.
[4] Y.Q. Cai, J.H. Wang, Differential evolution with hybrid linkage crossover, Inf. Sci. 320 (2015) 244–287.
[5] S. Das, A. Abraham, U.K. Chakraborty, A. Konar, Differential evolution using a neighborhood-based mutation operator, IEEE Trans. Evol. Comput. 13 (2009) 526–553.
[6] S. Das, S.M. Sankha, P.N. Suganthan, Recent advances in differential evolution—An updated survey, Swarm Evol. Comput. 27 (2016) 1–30.
[7] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (2011) 4–31.
[8] A. Draa, S. Bouzoubia, I. Boukhalfa, A sinusoidal differential evolution algorithm for numerical optimization, Appl. Soft Comput. 27 (2015) 99–126.
[9] M.G. Epitropakis, D.K. Tasoulis, N.G. Pavlidis, V.P. Plagianakos, M.N. Vrahatis, Enhancing differential evolution utilizing proximity-based mutation operators, IEEE Trans. Evol. Comput. 15 (2011) 99–119.
[10] H.Y. Fan, J. Lampinen, A trigonometric mutation operation to differential evolution, J. Glob. Optim. 27 (2003) 105–129.
[11] L. Fisher, The Perfect Swarm: The Science of Complexity in Everyday Life, Basic Books, 2010 ReadHowYouWant.com.
[12] J.C. Glenn, Collective intelligence: one of the next big things, Futura 28 (2009) 4.
[13] W. Gong, Z. Cai, Differential evolution with ranking-based mutation operators, IEEE Trans. Cybern. 43 (2013) 2066–2081.
[14] S.M. Guo, C.C. Chang, P.H. Hsu, J.S. Hong, Improving differential evolution with successful-parent-selecting framework, IEEE Trans. Evol. Comput. 19 (2015) 717–730.
[15] H. Guo, Y. Li, J. Li, H. Sun, D. Wang, X. Chen, Differential evolution improved with self-adaptive control parameters based on simulated annealing, Swarm Evol. Comput. 19 (2014) 52–67.
[16] S.M. Islam, S. Das, S. Ghosh, S. Roy, P.N. Suganthan, An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization, IEEE Trans. Syst. Man Cybern. B Cybern. 42 (2012) 482–500.
[17] J. Lampinen, I. Zelinka, On stagnation of the differential evolution algorithm, in: Proceedings of the Sixth International Mendel Conference on Soft Computing, 2000, pp. 76–83.
[18] P. Lévy, Collective Intelligence, Plenum/Harper Collins, 1997.
[19] J.J. Liang, B.Y. Qu, P.N. Suganthan, A.G. Hernández-Díaz, Technical Report 201212, Computational Intelligence Laboratory, Zhengzhou UniversityNanyang Technological University, Zhengzhou, ChinaSingapore, 2012.
[20] R. Mallipeddi, P.N. Suganthan, Q.-K. Pan, M.F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Appl. Soft Comput. 11 (2011) 1679–1696.
[21] R. Mallipeddi, G. Wu, M. Lee, P.N. Suganthan, Gaussian adaptation based parameter adaptation for differential evolution, in: Proceedings of the IEEE Congress on Evolutionary Computation, Beijing, China, 2014, pp. 1760–1767.
[22] T. O'Reilly, What is Web 2.0, O'Reilly Media, Inc, 2009.
[23] L. Page, S. Brin, R. Motwani, T. Winograd, The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.
[24] D.W. Palmer, M. Kirschenbaum, J. Murton, et al, Development of collective control architectures for small quadruped robots based on human swarming behavior, in: Proceedings of the Second International Workshop on the Mathematics and Algorithms of Social Insects, 2003, pp. 123–130.
[25] A.P. Piotrowski, Adaptive memetic differential evolution with global and local neighborhood-based mutation operators, Inf. Sci. 241 (2013) 164–194.
[26] I. Poikolainen, F. Neri, F. Caraffini, Cluster-based population initialization for differential evolution frameworks, Inf. Sci. 297 (2015) 216–235.
[27] G. Pór, Blog of collective intelligence, http://www.community-intelligence.com/blogs/public (2012).

[28] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2009) 398–417.
[29] P. Rocca, G. Oliveri, A. Massa, Differential evolution as applied to electromagnetics, IEEE Antennas Propag. Mag. 53 (2011) 38–49.
[30] N. Savage, Gaining wisdom from crowds, Commun. ACM 55 (3) (2012) 13–15.
[31] C. Segura, C.A. Coello Coello, A.G. Hernandez-Diaz, Improving the vector generation strategy of differential evolution for large-scale optimization, Inf. Sci. 323 (2015) 106–129.
[32] R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (1997) 341–359.
[33] J. Surowiecki, The Wisdom of Crowds, Anchor, 2005.
[34] T.M. Szuba, Computational Collective Intelligence, John Wiley & Sons, Inc, 2001.
[35] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, in: Proceedings of the IEEE Congress on Evolutionary Computation 2013, Cancún, México, 2013, pp. 71–78.
[36] L. Tang, Y. Dong, J. Liu, Differential evolution with an individual-dependent mechanism, IEEE Trans. Evol. Comput 19 (2015) 560–574.
[37] I. Triguero, S. Garcia, F. Herrera, Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification, Pattern Recognit. 44 (4) (2011) 901–916.
[38] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evol. Comput. 15 (2011) 55–66.
[39] Y. Wang, Z. Cai, Q. Zhang, Enhancing the search ability of differential evolution through orthogonal crossover, Inf. Sci. 185 (2012) 153–177.
[40] Y. Wang, H.-X. Li, T. Huang, L. Li, Differential evolution based on covariance matrix learning and bimodal distribution parameter setting, Appl. Soft Comput. 18 (2014) 232–247.
[41] J. Wang, J. Liao, Y. Zhou, Y. Cai, Differential evolution enhanced with multiobjective sorting-based mutation operators, IEEE Trans. Cybern. 44 (2014) 2792–2805.
[42] H. Wang, S. Rahnamayan, H. Sun, M.G.H. Omran, Gaussian bare-bones differential evolution, IEEE Trans. Cybern. 43 (2013) 634–647.
[43] X. Wang, L. Tang, An adaptive multi-population differential evolution algorithm for continuous multi-objective optimization, Inf. Sci. 348 (2016) 124–141.
[44] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1997) 67–82.
[45] G. Wu, R. Mallipeddi, P.N. Suganthan, R. Wang, H. Chen, Differential evolution with multi-population based ensemble of mutation strategies, Inf. Sci. 329 (2016) 329–345.
[46] M. Yang, C. Li, Z. Cai, J. Guan, Differential evolution with auto-enhanced population diversity, IEEE Trans. Cybernet. 45 (2015) 302–315.
[47] W.J. Yu, J.J. Li, J. Zhang, M. Wan, Differential evolution using mutation strategy with adaptive greediness degree control, in: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation GECCO'14, Vancouver, BC, Canada, 2014, pp. 73–79.
[48] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (2009) 945–958.
[49] Y. Zhang, C. Yu, W. Liu, W. Bi, Spectrum parameter estimation in Brillouin scattering distributed temperature sensor based on cuckoo search algorithm combined with the improved differential evolution algorithm, Opt. Commun. 357 (2015) 15–20.
[50] S.-Z. Zhao, P.N. Suganthan, Empirical investigations into the exponential crossover of differential evolutions, Swarm Evol. Comput. 9 (2013) 27–36.